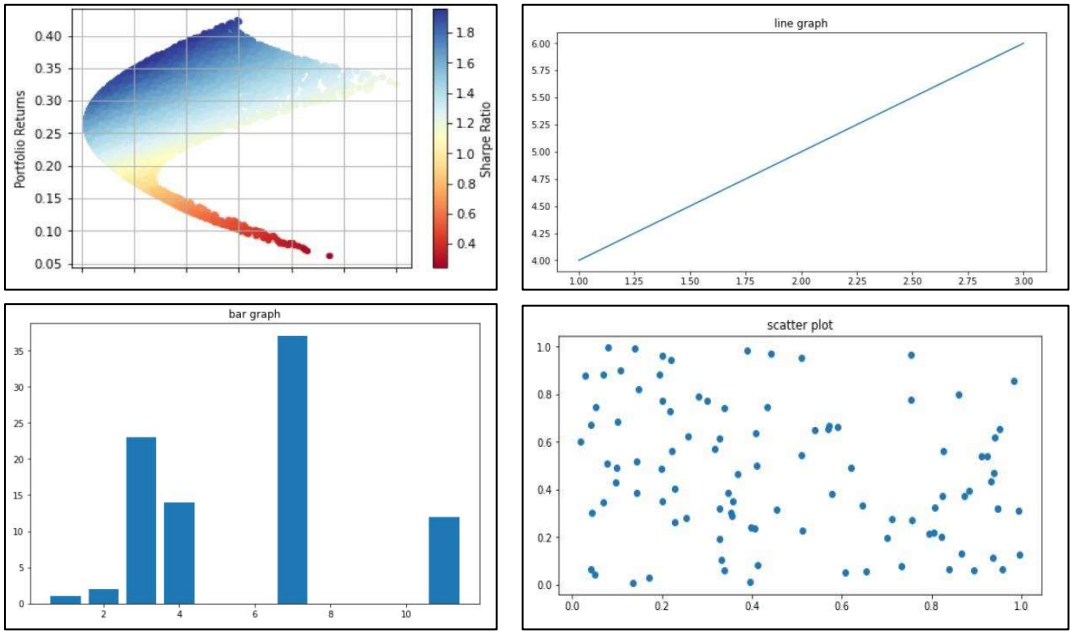
DATA VISUALIZATION USING PYTHON

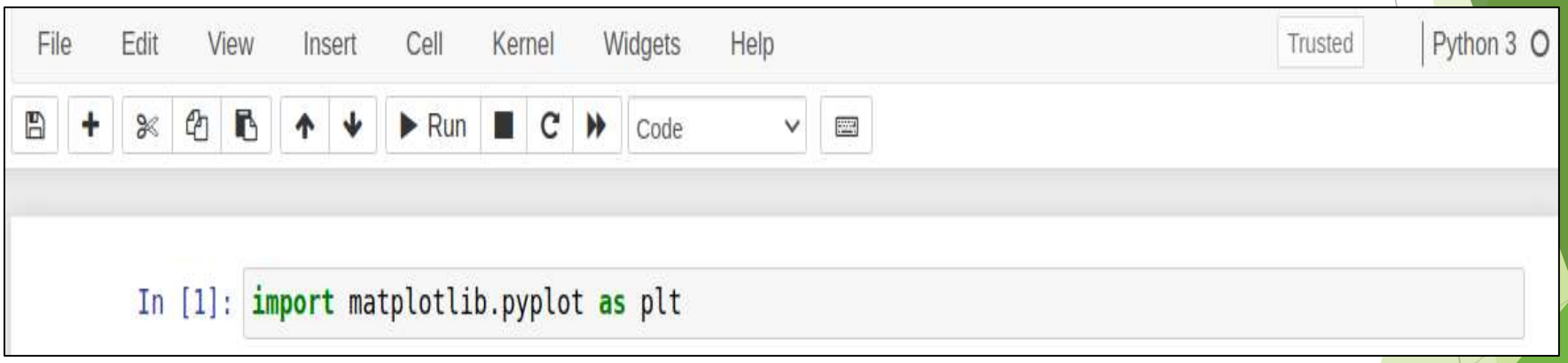
About Matplotlib

- Matplotlib is the most popular python library for plotting different kinds of graphs.
- The Pyplot module inside the Matplotlib makes it work like Matlab.



Importing Matplotlib

- To import matplotlib.pyplot, type ‘import matplotlib.pyplot’ in Jupyter Notebook and run the cell.
- The common abbreviation used for matplotlib.pyplot is plt.



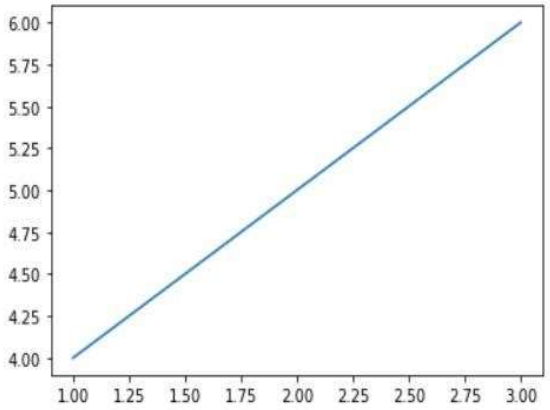
Plotting Line Plots (1/2)

- We can plot line plots using matplotlib using the .plot() function.
  - The first argument in the .plot() function specifies the x-axis.
  - The second argument in the .plot() function specifies the y-axis.

```
In [3]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.plot(x_axis, y_axis)

Out[3]: [<matplotlib.lines.Line2D at 0x7f186e0faf70>]
```



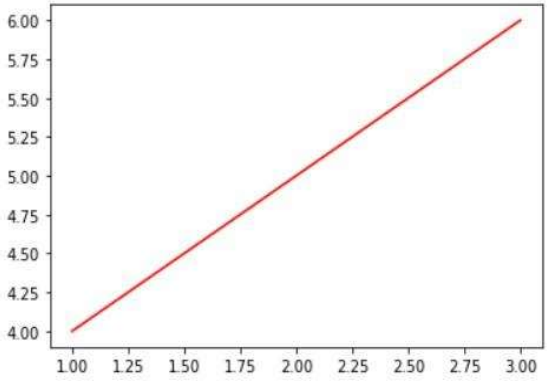
Plotting Line Plots (2/2)

- Changing Color
  - We can also change the color of the line by providing the color as third argument in the
  - plot() function.
  - A list of the color abbreviations can be found at: https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html

```
In [4]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.plot(x_axis, y_axis, 'r')

Out[4]: [<matplotlib.lines.Line2D at 0x7f186d85eb80>]
```



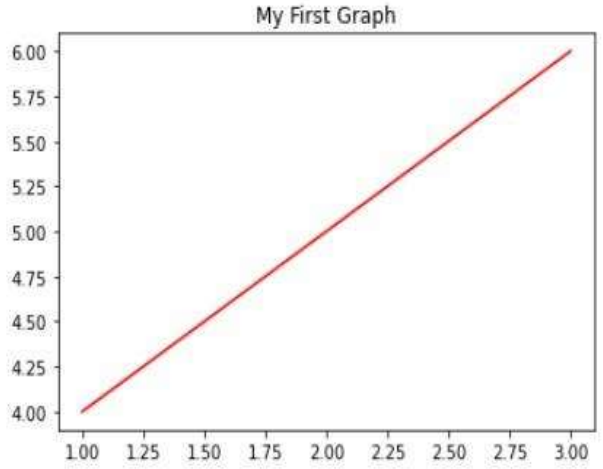
Title

- To set the title of the plot, use the .title() function.

```
In [4]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.title('My First Graph')
        plt.plot(x_axis, y_axis, 'r')

Out[4]: [<matplotlib.lines.Line2D at 0x7efef8336700>]
```



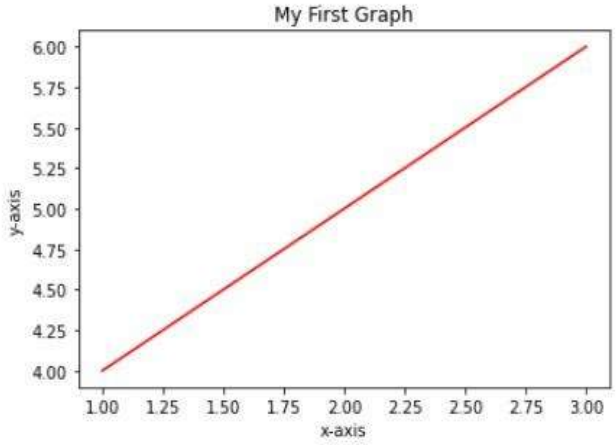
Labels

- To assign labels to x and y-axis, use .xlabel() and .ylabel() respectively.

```
In [5]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.title('My First Graph')
        plt.xlabel('x-axis')
        plt.ylabel('y-axis')
        plt.plot(x_axis, y_axis, 'r')

Out[5]: [<matplotlib.lines.Line2D at 0x7efef82a4340>]
```



Legend (1/2)

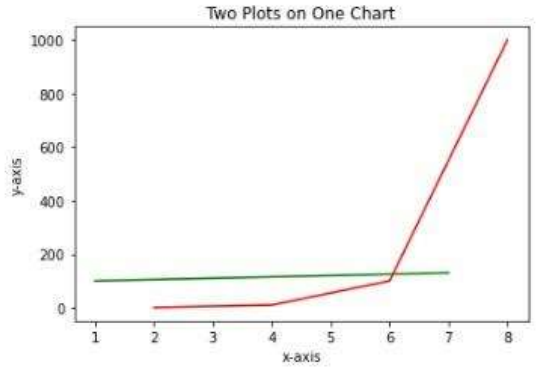
- We can plot multiple plots on the same chart simply by plotting them one by one as in the given example.

```
In [10]: x1_axis = [2, 4, 6, 8]
         y1_axis = [1, 10, 100, 1000]
         x2_axis = [1, 3, 5, 7]
         y2_axis = [100, 110, 120, 130]

         plt.title('Two Plots on One Chart')
         plt.xlabel('x-axis')
         plt.ylabel('y-axis')

         plt.plot(x1_axis, y1_axis, 'r')
         plt.plot(x2_axis, y2_axis, 'g')

Out[10]: [<matplotlib.lines.Line2D at 0x7efef81c7850>]
```



Legend (2/2)

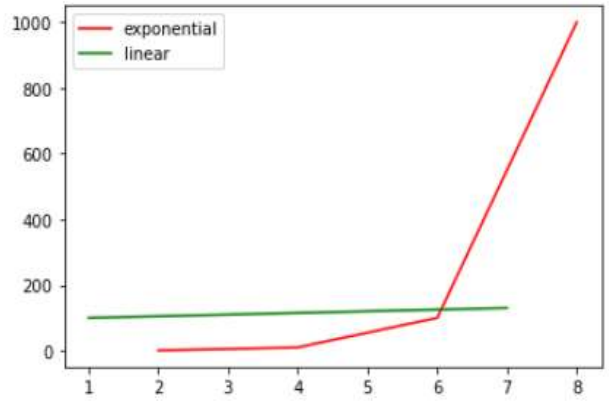
- If plotting more than one plots, it is a good idea to add a legend in your figure using the .legend() function.

```
In [12]: x1_axis = [2, 4, 6, 8]
         y1_axis = [1, 10, 100, 1000]
         x2_axis = [1, 3, 5, 7]
         y2_axis = [100, 110, 120, 130]

         plt.plot(x1_axis, y1_axis, 'r')
         plt.plot(x2_axis, y2_axis, 'g')

         plt.legend(['exponential', 'linear'])

Out[12]: <matplotlib.legend.Legend at 0x7efef81037c0>
```

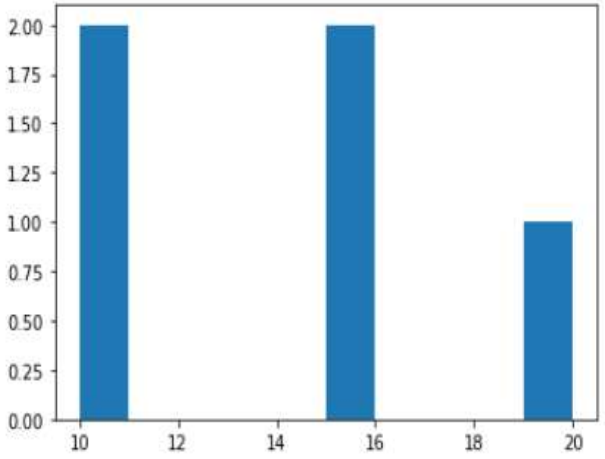


Plotting Histograms (1/3)

- We can also plot histograms using .hist() function.
- A histogram is generally used to plot frequency which helps identify distribution of data.

```
In [20]: values = [10, 15, 20, 10, 15]
         plt.hist(values)

Out[20]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```



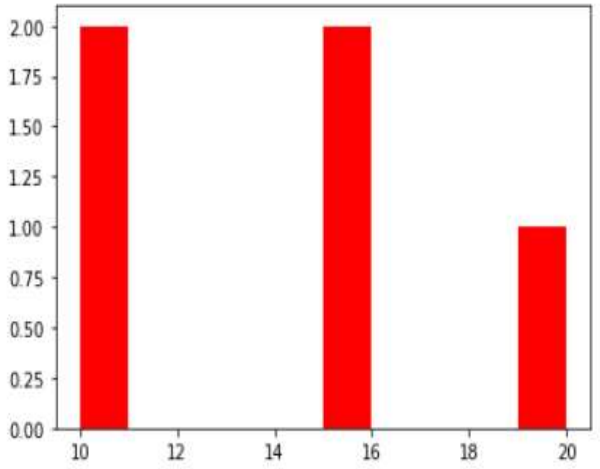
Plotting Histograms (2/3)

Changing Color

- We can also change color of the bars using the 'color' parameter inside the hist() function.

```
In [21]: values = [10, 15, 20, 10, 15]
         plt.hist(values, color='r')

Out[21]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```



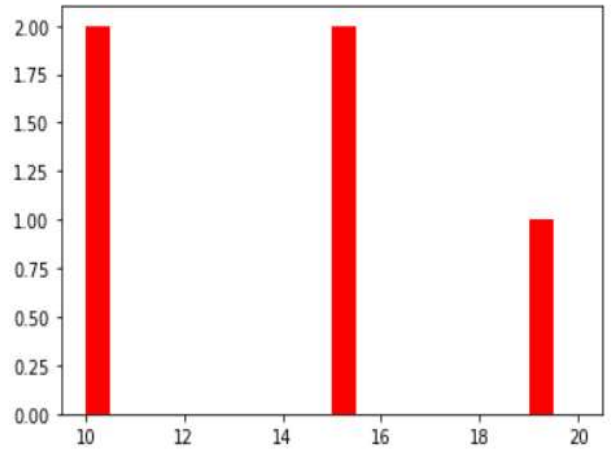
Plotting Histograms (3/3)

Changing Width

- We can also change width of the bars using the 'width' parameter inside the hist() function.

```
In [26]: values = [10, 15, 20, 10, 15]
         plt.hist(values, color='r', width=0.5)

Out[26]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```

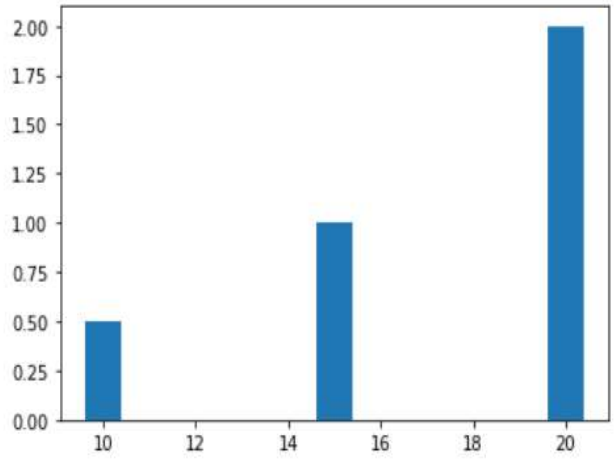


Plotting Bar Charts (1/2)

- To plot a bar graph, use the .bar() function of the matplotlib.pyplot.
  - First argument in the bar() function is the x-label.
  - Second argument in the bar() function is the height of each bar, which can be a list of values or a single value.

```
In [33]: values = [10, 15, 20]
         plt.bar(values, [0.5, 1, 2])

Out[33]: <BarContainer object of 3 artists>
```



Plotting Bar Charts (2/2)

Changing Width

- We can also change the width of bars in the bar plot using the 'width' parameter.

```
In [34]: values = [10, 15, 20]
         plt.bar(values, [0.5, 1, 2], width=2)

Out[34]: <BarContainer object of 3 artists>
```